A Comparison of User Experience, Interfaces, and Interaction in Computer Applications

An Honors Thesis (HONR 499)

by

Brandon Groff

Thesis Advisor

Stuart Sipahigil

Ball State University

Muncie, Indiana

April 2017

Expected Date of Graduation

May 2017

Abstract

As computers and applications have evolved to be used by vast populations, user experience and user interface have become central considerations in the development of software in order to accommodate users' expectations and needs. This study focused on the comparison of an older DOS application (DOS Goldman) and a newly developed native, web-based application (NeuroLab). Ball State University Physiology 215 students used the applications as part of an educational lab, and measurements of the amount of explanation required for use and the ease of navigation of the applications were recorded by survey. The results showed that the DOS Goldman application required more explanation and was more difficult to navigate than NeuroLab. These results were extrapolated to conclude that considerations of user experience and user interface are critical when designing software applications since today's users have higher expectations of software and its usability.

Acknowledgements

I'd like to thank Dr. Tucker for all of her cooperation and aid in the design of this application. I also need to thank her GAs, for giving me time during labs and allowing me to intrude on their teaching time. I also need to thank the Digital Corps, for giving me the base knowledge and confidence to take on the development of NeuroLab. Also, thanks to Stuart, my advisor, for helping me along with articles and advice on User Experience. Lastly, I need to thank my family for helping me get to this point in my life, and reassuring me that I've made the right decisions.

Table of Contents

Table of Figures

Process Analysis

Creating NeuroLab was not an idea entirely of my own invention. The application does not perform any new functions, nor did I even know what the Nernst-Goldman equation was before this project. I got the idea for this project from two directions. On one hand, through my work and personal experience, I became interested in application design. This led me to discover and research user experience. On the other hand, the idea for recreating an old application came from a close friend who took Dr. Tucker's physiology course that previous semester, and she spoke of how difficult to use the original application was. With both a research topic and a goal, I met with Dr. Tucker and began planning NeuroLab.

The developmental phase of the application required more time than the research phase. Many hours were spent designing, building, rebuilding, and compiling the application. In the end the application met all expectations, and it could be used in the classroom and for data collection.

Prior to conducting surveys with users, time was spent looking up articles on user experience and how to gather data that would answer questions about what the user expected. This led to many blogs, and my advisor assisted as well. What I learned is that sometimes asking the user the question you want an answer to is not the correct approach. Sometimes it takes a simple example question, and from that response the researcher can determine what the user needs. This aligns with the general concept of user experience, which is that the user knows what they are trying to do, but cannot always translate that to what they want or need.

Looking back, the project taught me more than I initially realized. One of the greatest rewards I have as a software developer is seeing users enjoy using the application you worked

so hard to create. I not only learned about a new framework and furthered my skills with web development and user experience, but I learned about the application design process, how to communicate with a client, how to lookup research, and how to conduct my own. It made me realize that while formal research is not my greatest strength or my specific career focus, I was able to use it to help my application succeed in a production setting.

In the end, I collected great data to write this thesis and saw the smiles on students' faces as they used NeuroLab. It's also satisfying to know that future students will have a better experience when using the application.

Introduction

As computers have progressed over time, the design and development of software applications has also had to progress to appeal to and be usable by the everyday user. In the physiology department at Ball State University, there is a lab that focuses on teaching the Goldman equation. This equation is used to calculate the membrane potential of ions crossing the cell membrane. In labs, this equation is used in an old computer application that had been developed before the turn of the millennium. This application is referred to as the DOS Goldman application because of the DOS operating system for which it was originally developed. The application is outdated and is not considered user-friendly. For this study, a new application, NeuroLab, was built to perform the same function as DOS Goldman, in a more modern, user-friendly fashion. To assess the potential improvements in usability of the application, a survey was conducted where the physiology students were asked to conduct the membrane potential lab on both the DOS Goldman application and the NeuroLab application and record their responses which were then quantified and compared.

When computers were first developed and applications for them were first built, they were quite basic compared to today's standards, and only used what are today taught as the basic building blocks of software programming. Those concepts include the command-line interface and the graphical user interface.

Interfaces

The original Command-Line (sometimes called *shell*) was created around 1963/64 by computer scientists at MIT [2]. The concept was simple: reduce time rewriting commands to do similar tasks. Rather than writing the same commands over and over, commands should be

used as building blocks for creating more complex commands [2]. With inspiration from another scientist's idea, Louis Pouzin wrote an article on this idea, coining it the *shell*. Around 1965, a MIT (Massachusetts Institute of Technology) graduate and GE (General Electric) man brought the shell to life [2]. While other problems arose, the shell slowly evolved into the modern-day terminal, and thus formed the backbone of the command-line interface.

A Command-Line Interface (CLI) is a text-based computer interface used to run and interact with command-line software. The CLI is "an interactive system where user input is achieved through lines of text" [1]. It is sometimes also referred to as a Console User Interface (CUI), Command Language Interpreter, and Character user interface, but no matter what it is called, the CLI is one way that allows a computer user to interact with computer software.

Historically, the CLI was the only way to interact with a computer. Today, the CLI is more commonly used by programmers and advanced computer users because of its less natural interface. In many cases, the CLI is more powerful than graphical user interfaces and more versatile. It can be used to perform basic commands like file moving and editing to more complex commands like software compiling and using build tools. The CLI can also serve to automate computational tasks by creating scripts, or a list of CLI commands to run. In this way, the CLI is sometimes viewed as having its own programming language. Common examples of the CLI include Windows DOS, the Windows CMD prompt, the BASH shell, and the Mac Terminal.

As opposed to a CLI, a Graphical User Interface (GUI), or sometimes just User Interface (UI), is, "an interface between a user and a computer system that makes use of input devices other than the keyboard and presentation techniques other than alphanumeric characters." [3]

These interfaces consist of windows, panes, icons, menus, pointing devices, and more. Most of these visual items can also be moved around the screen, scaled, minimized/expanded, and closed. Contrary to the CLI, which is primarily navigated with a keyboard, a GUI is primarily navigated by pointing methods (a mouse or touch input), but it can also use the keyboard if shortcut commands are built into the GUI. Examples of common GUIs include the Windows 7 operating system, the MacOS/OS X operating system interface, and websites. The basics of computer application development have evolved to use these foundational elements of user interfaces, plus other considerations that help make applications more user-friendly.

Today, developers focus on user experience, user interface, and human-computer interaction as the pillars of application development. User experience is, "the experience the product creates for the people who use it in the real world" [10]. User experience in computing walks a fine line between software development and creative design as it takes user ideas, uses the design process to generate the best designs, and translates them into developmental terms.

A User Interface (UI) is an interface that makes use of input devices to allows a computer user to interact with the computer. While many people use the term "user interface" to refer to GUI, a UI can refer more generically to any interface that allows communication between user and computing system. For this reason, it should be noted that both a CLI and a GUI are considered UIs. For clarity's sake, this study will always specify CLI or GUI when talking about one or the other, and UI when referring to the more generic umbrella of all user interfaces.

Human-Computer Interface (HCI), or sometimes Human-Computer Interaction, is referred to as a, "means of communication between a human user and a computer system,

referring in particular to the use of input/output devices with supporting software" [4]. In the past, this referred to a simple monitor or printer in combination with a keyboard. With advancements in technology, this list has increased to include all types of visual output, touch-sensitive devices, and voice-input technology. Simply, this list includes anything that allows a user to communicate with a computing device.

In order to develop complex applications that match the goals of the user, programming has changed over time to include various languages that each run in different ways based on the type of application being made and the operating system.

By definition, programming languages are "artificial languages in which syntax and semantics are strictly defined" [12]. These languages form the backbone of all software programs. In the early days of computing, computer instructions were made via physical punch cards. As computing shifted from mechanical to digital processes, assembly languages took the floor. These assembly programming languages required a large intellectual effort to write and were error prone. Thus, programming languages were created to replace them with more abstract methods than directly programming the processor. Some of the earliest and most well-known programming languages include Fortran (1957), Lisp (1958), COBOL (1959), and BASIC (1964). These early languages did not have the best-defined syntax and semantics, but they worked well and formed the building blocks of more modern languages like C (1972), Prolog (1972), and SQL (1978). The C programming language, despite numerous language and compiler updates over the years, is still one of the most popular programming languages because of its direct memory and hardware level control. Between C and SQL (Structured Query Language), many languages since have been created based off their core concepts. Popular languages since

then include C++ (1980), Python (1991), Ruby (1993), Java (1995), JavaScript (1995), PHP (1995), and C# (2001). These languages have all found common uses in today's digital society, including the NeuroLab application.

Operating Systems

An Operating System (OS) is the core software system that directly manages interaction between user interface devices and system hardware. As part of this, the OS "jointly controls the system resources and the processes using these resources on a computer system" [16]. Examples of OSes include DOS, Windows, OS X (or MacOS), and Linux.

Certain applications are built to only be compatible with a specific OS or platform, and these are called "native" applications. Native applications usually provide the greatest hardware control and performance for applications, but they are not useable across every device/platform. Developing a web application allows for more device versatility. A web application is an application built using web technologies (like HTML, CSS, and JavaScript) that can only run in a web browser. The NeuroLab application was built as a native application with web technologies, so it can be used on Windows, Mac, or Linux operating systems.

Computer applications can only be built to the maximum capability of the computer and OS on which they run. The DOS Goldman application was built around the late 1980s/early 1990s, a time when computer hardware and graphics were limited, and most computer users were professionals or tinkerers rather than everyday users. Comparatively, today's computers are used by people of all backgrounds and experiences, and the hardware and graphics are hundreds of times more powerful. NeuroLab was built with the common user and today's advanced technology in mind, so the development was based on input from user experience,

UI, and HCI. With this knowledge, the expected result of the study was that users would require less explanation to use the NeuroLab application, and the NeuroLab application would demonstrate easier navigation of the application. Specifically, NeuroLab should succeed because it takes advantage of a colorful point-and-click interface whereas the DOS Goldman application is limited to a black and white, keyboard-only interaction. Such results would support the importance of considering user experience and user interface when building applications.

Methods

Application

Requirements

In developing NeuroLab, there was only one functional requirement to be met. Because this program was designed to replace another application, it needed to be able to correctly perform the Nernst-Goldman equation. Beyond this, all other requirements were non-functional, but equally important. These requirements included the ability to run on Windows OS and to provide a simple interface capable of being used in an educational setting since the study was conducted at Ball State University in a Physiology lab.

Design Process

As a part of the design process, the first step in the creation of NeuroLab was conceptualizing the base needs of the application. This consisted of figuring out the end objective of the application and designing an interface that would meet those needs. In this case, the end objective was to perform the Nernst-Goldman function and show users the output. Thus, a simple form was designed to include the input fields required to perform this function and demonstrate the equation including the desired inputs.

With an initial design sketched up, it was ready for review. A former Physiology 215 student and other peers reviewed the mockups. The professor of the Physiology lab in which the application was used also reviewed the design. Revisions were made to the design, and the final application was built from them.

Development

NeuroLab was designed as a native desktop application using web technologies. Specifically, NeuroLab was made using HTML, SASS (CSS preprocessor), and AngularJS, a JavaScript framework that focuses on a Model-View-Controller (MVC) developmental style. Another tool called Electron was also used as a web application wrapper, which allows web applications to be placed into a native desktop application.

NeuroLab was built using web technologies for a couple of reasons, the largest of which was the developer's familiarity with web technologies. Another benefit of using web technologies with Electron is cross-platform compatibility. While the requirements for this application only dictated Windows OS compatibility, Electron allowed NeuroLab to be built once and exported for use for Windows OS, OS X/Mac OS, and even Linux builds.

Code

Below is an excerpt of some code used to write the NeuroLab application. The following URL provides access to the repository used to store and host the application code: https://github.com/EPICmynamesBG/NeuroLab

```
app.controller("GoldmanController", function GoldmanController($scope,
$location, $rootScope, $http, $timeout) {
    require('./js/jquery/scroll-fix.js');

    // Base model for saving default values. Currently statics, may be
    // expanded in the future
    var settingsManager = new SettingsManager();
```

```
// Electron process communicator
// Handles menu bar item clicks
const ipcRenderer = require('electron').ipcRenderer;
// Used to throw open the results window
const BrowserWindow = require('electron').remote.BrowserWindow;
var resultsWindow = null;


// Menu: Form > Calculate
ipcRenderer.on('calculate', (event, message) => {
    $scope.calculate();
});
// Menu: Form > Clear
ipcRenderer.on('clear', (event, message) => {
    $scope.clear();
    $scope.$digest();
});

// Update menubar enable/disable for calculate as form validity
changes
$scope.$watch('NG_form.$invalid', function (newValue) {
    ipcRenderer.send('setCalculate', !newValue);
});
// Update menubar enable/disable for reset as form validity changes
$scope.$watch('NG_form.$pristine', function(newValue) {
    ipcRenderer.send('setClear', !newValue);
});

// Load default/saved settings
```

```
        function loadSettings() {
            var settings = settingsManager.getSettings();
            $scope.NG_RT = settings['NG-RT'];
            $scope.z = settings['NG-z'];
        }
        loadSettings();

        // Shows the calculate popup window
        function showCalculatePopup() {
            var baseURL = 'file://' + __dirname +
'/html/goldmanCalculation.html';
            resultsWindow = new BrowserWindow({
                width: 450,
                height: 450,
                show: false,
                alwaysOnTop: true,
                resizable: false,
                icon: '../images/icon.ico'
            });
            resultsWindow.on('closed', function () {
                resultsWindow.show = false;
                resultsWindow = null;
            });
            //url params: NG_RT, NG_z, KIn, KOut
            var parameters = "?NG_RT=" + $scope.NG_RT +
                "&NG_z=" + $scope.z +
                "&KIn=" + $scope.potassiumInside +
                "&KOut=" + $scope.potassiumOutside +
                "&NaIn=" + $scope.sodiumInside +
```

```
            "&NaOut=" + $scope.sodiumOutside +
            "&KPerm=" + $scope.potassiumPerm +
            "&NaPerm=" + $scope.sodiumPerm;

        parameters = encodeURI(parameters);

        resultsWindow.loadURL(baseURL + parameters);
        resultsWindow.setMenu(null);
        resultsWindow.show();
    }

    // Click event. Handles the calculate popup window, ensuring
    // more than one instance is not created
    $scope.calculate = function () {
        if (resultsWindow == null) {
            showCalculatePopup();
        } else {
            resultsWindow.close();
            if (resultsWindow != null) {
                resultsWindow.destroy()
            }
            showCalculatePopup();
        }
    }

    // Clears all the current input values
    $scope.clear = function () {
        $scope.NG_RT = null;
        $scope.z = null;
```

```
            $scope.potassiumInside = null;
            $scope.potassiumOutside = null;
            $scope.sodiumInside = null;
            $scope.sodiumOutside = null;
            $scope.potassiumPerm = null;
            $scope.sodiumPerm = null;
            $scope.NG_form.$setPristine();
            loadSettings();
        }

    });
```

Survey

While NeuroLab was built to perform the Nernst-Goldman calculation, it was also used to provide insight on how users can effectively use an application. A questionnaire was created that would compare the usability of the older application and NeuroLab. The questionnaire was short, but designed with a simple goal in mind: to determine if the experience using the application was were different from the old to the new version, while performing the same task in both applications.

This survey was given to around 60 students in Professor Tucker's Physiology lab at Ball State University. All students had basic knowledge of the Nernst-Goldman equation and the theory behind it. The survey featured the same five questions for both the DOS Goldman application and NeuroLab. Space was also left for students to leave comments about usability of the applications.

Analysis/Statistics

The data was transferred to an Excel sheet for analysis. In Excel, the multiple-choice responses were graphed. Each response was assigned a number so that the data could be statistically analyzed. The "Amount of Explanation" question responses, none, a little, some, and a lot, were assigned numbers 1, 2, 3, 4, respectively. The "Ease of Navigation" question responses, very easy, easy, average, difficult, very difficult, were assigned numbers 1, 2, 3, 4, 5, respectively. The data was averaged and standard deviation was calculated. Short response questions were manually analyzed to determine response patterns.

Results

The sample size was 61 students. The average amount of explanation required for use of the DOS Goldman application was 3.3 ± 0.8, or "some" explanation (Figure 1). The average amount of explanation required for the NeuroLab application was 1.1 ± 0.5, or "none" (Figure 2). The average ease of navigation for the DOS Goldman application was 4.0 ± 0.7, or "difficult" (Figure 3). The average ease of navigation for the NeuroLab application was 1.1 ± 0.3, or "very easy" (Figure 4).

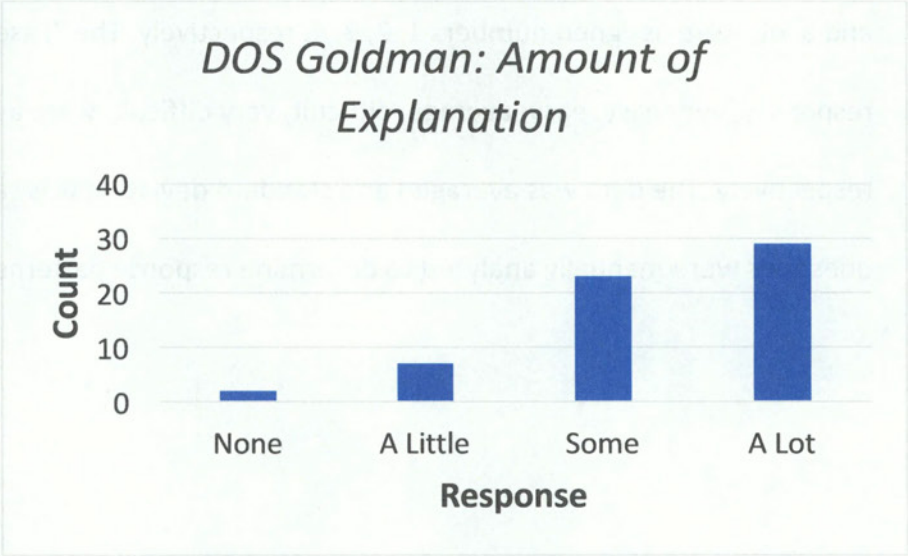

Figure 1. *Amount of explanation required for DOS Goldman application use.*

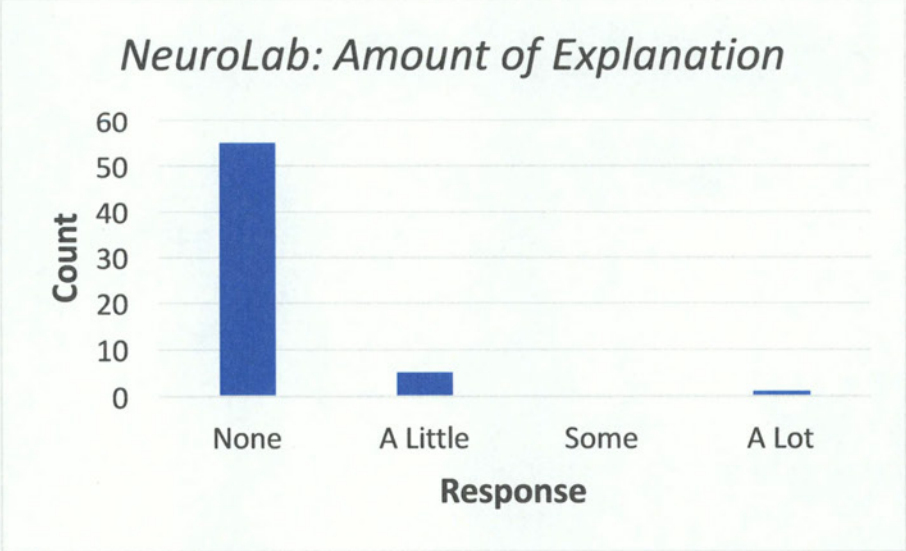

Figure 2. *Amount of explanation required for NeuroLab application use.*

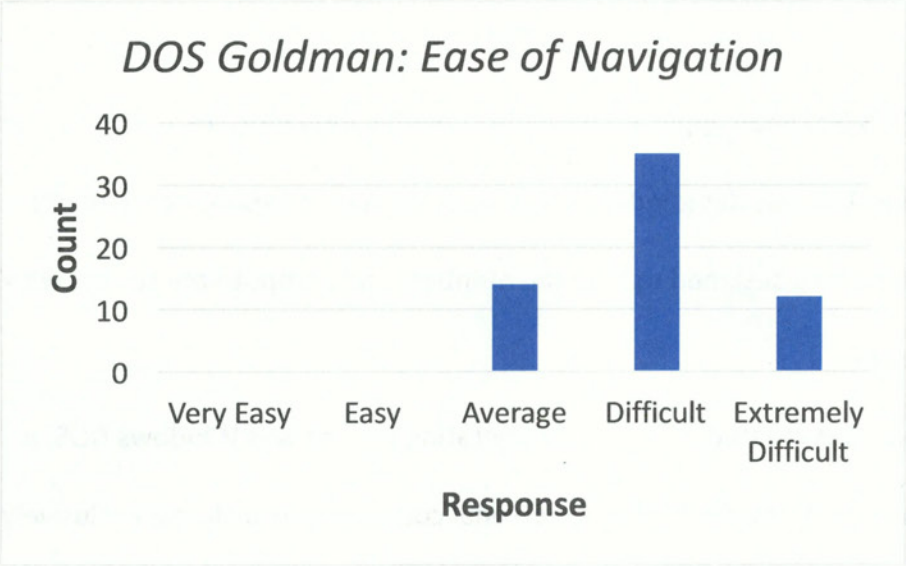

Figure 3. *Ease of navigation of DOS Goldman application.*

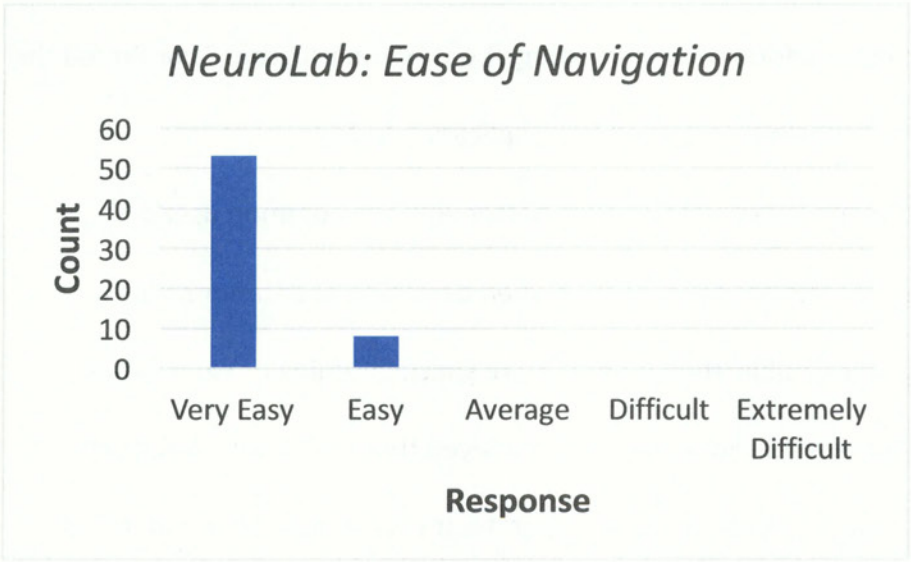

Figure 4. *Ease of navigation of NeuroLab application.*

Discussion

As expected, the DOS Goldman application required significantly more explanation prior to use than NeuroLab. NeuroLab was designed and developed with user experience in mind; comparatively, DOS Goldman was designed to take the numbers and compute the results with no regard for user experience.

When DOS Goldman was created, the primary operating system was Windows DOS, a command-line only operating system. At that time, personal computing was almost exclusively command-line based and had not advanced to the GUI. Because of this, software was primarily developed with little regard to user expectations. Comments from the survey such as, "I had trouble figuring out how to input information and getting it out of the program," confirmed the need for additional instruction to use a command-line application today.

Today, computer software is expected to align better with the common user's expectations; hence, these expectations must be met when designing applications. An application for use by the general public should not require special training or explanation. NeuroLab was designed to require no explanation and achieved this goal. Users specifically commented that, "no instructions [were] needed to operate. It was straight to the point." The comments and data supported the hypothesis that NeuroLab would require less explanation than DOS Goldman.

Since the advent of personal computing, applications have been driven to use a design that is intuitive even to non-technical users. For this to be achieved, software had to be conscientiously designed and developed to require minimal training for use. This transition in software design began with XEROX Company, which developed the graphical interface and

mouse to select operations for their copier products. These GUI concepts were then reused by Apple in their original GUI and became a standard for personal computing devices [13]. These point-and-click machines and software set the standard for usability; a standard that still applies today, and was embedded into the core of NeuroLab's design.

NeuroLab's ease of navigation was rated as 'very easy'. DOS Goldman was difficult for users to navigate because it is only navigable via keyboard; the keyboard inputs for editing a field are not obvious, and there are more configurable values than necessary (Figure 5). Comments about DOS Goldman's ease of navigation included: "changing the numbers and knowing what keys to use was hard" and "it [was] hard to understand which keys do which function." Comments like these reaffirm the challenges posed by a poor UI.

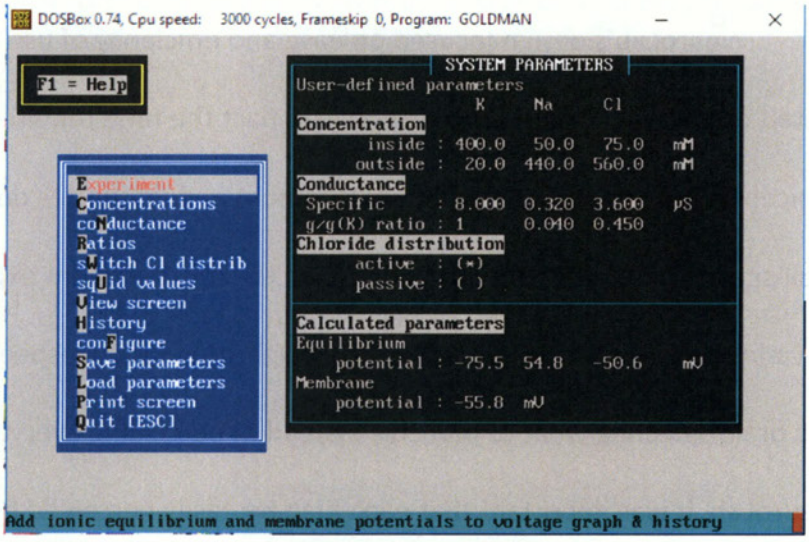

Figure 5. DOS Goldman interface.

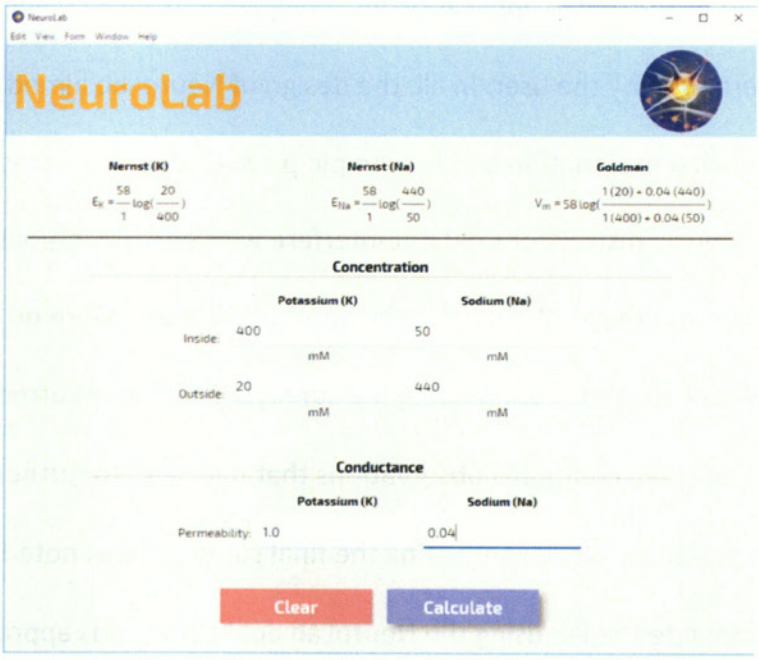

Figure 6. NeuroLab interface.

Comparatively, users found NeuroLab's interface to be obvious

because, "it told you what & where to enter information." In another user comment, NeuroLab was easier to navigate "because it wasn't so cluttered" (Figure 6). When designing NeuroLab, major consideration was given to input fields because poorly labelled inputs distract users and create a barrier for accomplishing the task users perform. Also, keeping the interface well-spaced and simple was important because a cluttered interface generates confusion.

NeuroLab's design focused on ease and efficiency of use of the application. Studies have noted that visual distractions negatively impact the time required to complete a task[14]. This concept is applied to application development and influences developers to keep applications well-spaced and uncluttered so users have the most efficient experience possible. Other studies have found that high perceptual load, or high stimulation, especially in the visual field, leads to the brain selecting what to perceive instead of perceiving everything[15]. In developing NeuroLab, it was important that each field be easily perceived by the user so they could be properly filled to perform the function. Hence, the design of NeuroLab kept perceptual load low so that the whole application including all of its entry fields and function buttons were perceived by the user. In all, the design of NeuroLab aimed to not only be easy to use with minimal explanation and use simple point-and-click navigation, but it also aimed to be visually simple so distractions did not interfere with end users' goal.

In the final design of the application, there were no errors in the functionality of NeuroLab, and in conducting the survey, no errors occurred. However, changes could be made to quantify additional observations that may lead to further analysis of the difference in user experience. While conducting the final survey, it was noted that the time for students to complete the lab using the NeuroLab application was approximately half of the time needed for

them to complete the lab using the DOS Goldman application. This was an anecdotal observation and was not recorded or officially quantified, but formal observations about time to complete a task using various applications could be well-applied to such a study.

Since NeuroLab was designed for an educational setting, it could also be beneficial to consider recording observations about learning improvement. This was not the goal of the study, but it could be an additional avenue of research to pursue when considering user experience, UI, and HCI in the context of technology in the classroom. Some students who completed the survey left comments directly related to the improved learning experience when using NeuroLab: "it showed the equations... which is helpful in learning," indicating a positive learning experience as a result of the user-conscious design. Again, the goal of the study was not to compare the learning experiences between the applications so these observations were not quantified, but clearly differences do exist.

A final important outcome of this study was the recognition that there is always room for improvement, even in the best designed applications. Everyone has their own opinions, and while most users left comments that the NeuroLab application was "great," a few left suggestions for design modifications that they thought would improve usability. Because of the differing opinions and usage styles of individuals, some people make recommendations for user experience, UI, or HCI changes that others would not appreciate or agree with. It is important as a developer and designer to take into consideration the application's audience in order to create the optimal solution. However, recommendations are taken seriously, which is why applications are often updated over time with various modifications that improve user experience, UI, and HCI.

Conclusion

The comparison of NeuroLab and DOS Goldman showed that while both applications performed the same task, users had a far better, easier experience using NeuroLab. Research showed that user navigation was simpler, and difficulty obtaining the desired results from the application was reduced. The improved experience was directly attributed to the user-conscious design of NeuroLab that reduced perceptive load and was designed for users of variable experience levels, compared to DOS Goldman's older design targeted for technically skilled users only.

Sources

1. A. Butterfield, G. E. Ngondi. (2016). "command-line interface (CLI)". *A Dictionary of Computer Science.* (7th Edition). [On-line]. Available: http://www.oxfordreference.com.proxy.bsu.edu/view/10.1093/acref/9780199688975.001.0001/acref-9780199688975-e-866?rskey=osNlQC&result=992 [Jan 5, 2017].

2. L. Pouzin. "The Origin of the Shell." Internet: http://www.multicians.org/shell.html, Nov.25, 2000 [Jan. 18, 2017].

3. A. Butterfield, G. E. Ngondi. (2016). "graphical user interface (GUI)". *A Dictionary of Computer Science.* (7th Edition). [On-line]. Available: http://www.oxfordreference.com.proxy.bsu.edu/view/10.1093/acref/9780199688975.001.0001/acref-9780199688975-e-2244 [Jan 6, 2017].

4. A. Butterfield, G. E. Ngondi. (2016). "human-computer interface (HCI)". *A Dictionary of Computer Science.* (7th Edition). [On-line]. Available: http://www.oxfordreference.com.proxy.bsu.edu/view/10.1093/acref/9780199688975.001.0001/acref-9780199688975-e-2436?rskey=D8sv6l&result=2 [Jan 6, 2017].

5. M. McHugh. "Yes, there is a Difference Between 3D Touch and Force Touch." Internet: https://www.wired.com/2015/09/what-is-the-difference-between-apple-iphone-3d-touch-and-force-touch/, Sept. 9, 2015 [Jan. 18, 2017].

6. D. A. Boehm-Davis. "Discoveries and Developments in Human-Computer Interaction". *Human Factors*, vol. June 2008, pp. 560-564.

7. R. Gupta. "Human Computer Interaction – A Modern Overview". *Int. J. Computer Technology & Applications*, vol. 3, pp. 1736-1740.

8. S. Lee. "Understanding User Experience with Computer-Based Applications with Different Use Purposes". *Intl. Journal of Human-Computer Interaction,* vol. 29, pp. 689-701.

9. A. Butterfield, G. E. Ngondi. (2016)."user-friendly". *A Dictionary of Computer Science.* (7th Edition). [On-line]. Available: http://www.oxfordreference.com/view/10.1093/acref/9780199688975.001.0001/acref-9780199688975-e-5631?rskey=0jeuNd&result=8 [Jan 6, 2017].

10. J.J. Garrett. "User Experience and Why It Matters" in *The Elements of User Experience*, 2nd Edition. M. Nolan, Ed. United States: Peachpit, 2011, pp. 3-17.

11. G. Solomon. "From Coding to Coding: Programming to Software to Web Tools to Apps to Programming in Classrooms." Internet: http://www.techlearning.com/resources/0003/from-coding-to-coding/69562 , Aug. 25, 2015 [Jan. 25, 2017].

12. A. Butterfield, G. E. Ngondi. (2016). "programming language". *A Dictionary of Computer Science.* (7th Edition). [On-line]. Available: http://www.oxfordreference.com/view/10.1093/acref/9780199688975.001.0001/acref-9780199688975-e-4161?rskey=2rgiRk&result=1 [Jan 28, 2017].

13. (2002, Oct. 11). *Computer Science 210: Introduction to Information Systems and Computer Applications*. [Digital]. Available: http://ccis.athabascau.ca/html/courses/comp210/CourseSample/chap01/section2.htm [Feb. 8, 2017].

14. J. Gasper, J. McDonald. (2014, Apr.). "Suppression of Salient Objects Prevents Distraction in Visual Search." *The Journal of NeuroScience.* [Online]. 34(16). Available: http://www.jneurosci.org/content/34/16/5658 [Feb. 9, 2017].

15. H. Marciano, Y. Yeshurun. (2014, Oct.) "Perceptual Load in Different Regions of the Visual Scene and Its Relevance for Driving." *Human Factors: The Journal of the Human*

Factors and Ergonomics Society. [Online]. 57(4). Available: http://journals.sagepub.com/doi/full/10.1177/0018720814556309 [Feb. 9, 2017].

16. A. Butterfield, G. E. Ngondi. (2016). "operating system (OS)". *A Dictionary of Computer Science.* (7th Edition). [On-line]. Available: http://www.oxfordreference.com/view/10.1093/acref/9780199688975.001.0001/acref-9780199688975-e-3642?rskey=Q79bBi&result=1 [Feb. 9, 2017].

Appendix

User Experience: A Comparison and Generalization

Questionnaire

To be used for both legacy and updated application responses

1. How much explanation did this application require to understand how to use it?

None A Little Some A Lot

2. How easy was it to navigate this application?

Very Easy Easy Average Difficult Extremely Difficult

3. Please provide a short description to your answers above. (Why was/wasn't this application easy to understand and navigate?)

4. Did this application accomplish what it was made to do?

Yes No*

*** If No**, please elaborate on why this application did not accomplish what it was made to do

5. Final comments on the usability/experience of the application?